At higher levels there are problems with allowable vocabularies. Present systems attempt to employ vocabularies in which the words are well separated in a feature space. As vocabularies grow, however, or as the choice of words becomes constrained (by a task domain, for example), then the possible errors in matching can be expected to increase. At the syntactic level, it is questionable how much more progress can be achieved without the use of general grammars, as opposed to simple ad hoc grammars. In this regard, the interface between grammars of this type and the phonemic processing level is not yet well understood.

Semantic support is another problem area since many of the interesting applications of speech understanding do not lend themselves to precise semantic formulation. The spontaneity which is a major advantage to speech input works against an understanding system here.

From the hardware point of view, there remain the expected problems of real-time response, processing power, memory size, systems organization, and cost.

In summary, significant progress in speech understanding awaits developments in many areas. It is hoped, however, that many such developments will occur in the next few years.

References:

A. Newell et al "Speech Understanding Systems", North-Holland, Amsterdam, 1973.

D.H. Klatt and K.N. Stevens "Sentence Recognition from Visual Examination of Spectrograms and Machine-Aided Lexical Searching", Conference Record, 1972 Conference on Speech Communication and Processing, Newton, Mass., April 1972.

PLANNER

Central Ideas:

Planner is both a problem solving formalism and a programming language. It stresses the importance of goal-orientation, procedural representation of knowledge, pattern directed invocation of procedures and a flexible backtrack-oriented control structure in a problem solver and in a high level programming language.

Technical Description:

Planner was developed as a formalism for problem solving by Hewitt (1972,1973) and a subset of the Planner ideas was implemented by Sussman et al (1973) in a programming language called Micro-Planner.

Planner is primarily oriented towards the accomplishment of goals which can in turn, be broken down into multiple subgoals. A goal in this context can be satisfied by finding a suitable assertion in an associative data base, or by accomplishing a particular task. Multiple goals may be activated at the same time, as might occur, for example in a problem reduction type of problem solver. The attempt to satisfy a goal is analogous to an attempt to prove a theorem. Planner, however, is not strictly a theorem-prover. The differences are mainly due to the types of knowledge which it can manipulate.

The traditional theorem-prover accepts knowledge expressed in declarative form, as in the predicate calculus; that is, as statements of "fact" about some problem domain. Planner, by contrast, is able to deal as well with knowledge expressed in imperative form; that is, knowledge which tells the problem solver how to go about satisfying a subgoal, or how to use a particular assertion. In fact the emphasis in Planner is on the representation of knowledge as procedures. This is based on the view that knowledge about a problem domain is intrinsically bound up with procedures for its use.

The ability to use both types of knowledge leads to what has been called a hierarchical control structure; that is, any procedure (or theorem in Planner notation) can indicate what the theorem-prover is supposed to do as it continues the proof.

Procedures are indexed in an associative data base by the patterns of what they accomplish. Thus, they can be invoked implicitly by searching for a pattern of accomplishment which matches the current goal. This is known as pattern directed invocation of procedures, and is another cornerstone of the Planner philosophy.

The final foundation of Planner is the notion of a backtrack control structure. This allows exploration of tentative hypotheses without loss of the capability to reject the hypotheses and all of their consequences. This is accomplished by remembering decision points (that is, points in the program at which a choice is made) and falling back to them, in order to make alternate choices, if subsequent computation proves unsuccessful.

Example:

The following, somewhat hackneyed, but still illustrative example is described in pseudo Micro-Planner. We will assume that the data base contains the following assertions.

```
(HUMAN TURING)
(HUMAN SOCRATES)
(GREEK SOCRATES)
```

together with the theorem

```
(THCONSE (X) (FALLIBLE $?X)
             (THGOAL (HUMAN $?X)))
```

where the theorem is a consequent theorem which can be read as - if we want to accomplish a goal of the form (FALLIBLE $?X), then we can do it by accomplishing the goal (HUMAN $?X).

We now ask the question "is there a fallible Greek ?". This can be expressed as

```
(THPROG (X)
     (THGOAL (FALLIBLE $?X) $?T)
     (THGOAL (GREEK $?X))
     (THRETURN $?X))
```

This program uses a linear approach to answering the question; that is, it first attempts to find something fallible, then check that what it has found is Greek. Is so, it returns what it has found.

Consider what happens when this program is applied to the data base above. It first finds nothing that is fallible in the list of assert-ions, and hence tries the theorem, and searches again for something human. It finds (HUMAN TURING) and binds TURING to $?X. However, an attempt to prove (GREEK TURING) fails. At this point, the backtrack control structure comes into play. The program returns to the last point at which a choice was made; that is, to the point at which TURING was bound to $?X. This binding is undone and the data base is searched again for something human. This time (HUMAN SOCRATES) is found and SOCRATES is bound to $?X. An attempt to prove (GREEK SOCRATES) succeeds and SOCRATES is returned as the value of the THPROG.

This example illustrates, albeit superficially, the basic tenets of the Planner formalism as they apply in a programming language. The reader is encouraged to consult the references for the complete details.

References:

C. Hewitt, "Description and Theoretical Analysis (using schemas) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", Phd Thesis, MIT,Feb., 1971.

C. Hewitt, "Procedural Embedding of Knowledge in PLANNER", 2nd IJCAI, 1971.

G. J. Sussman, T. Winograd, and E. Charniak, "MICRO-PLANNER Reference Manual", MIT AI Memo 203A, December, 1971.

APPENDIX B

AI HANDBOOK OUTLINE

NOTE:
The following material describes work in progress and planned for publication. It is not to be cited or quoted out of the context of this report without the express permission of Professor E. A. Feigenbaum of Stanford University.

I. INTRODUCTION

A. Intended Audience

This handbook is intended for two kinds of audience; computer science students interested in learning more about artificial intelligence, and engineers in search of techniques and ideas that might prove useful in applications programs.

B. Suggested Style For Articles

The following is a brief checklist that may provide some guidance in writing articles for the handbook. It is, of course, only a suggested list.

i) Start with 1-2 paragraphs on the central idea or concept of the article. Answer the question "what is the key idea?"

ii) Give a brief history of the invention of the idea, and its use in A.I.

iii) Give a more detailed technical description of the idea, its implementations in the past, and the results of any experiments with it. Try to answer the question "How to do it?.

iv) Make tentative conclusions about the utility and limitations of the idea if appropriate.

v) Give a list of suitable references.

vi) Give a small set of pointers to related concepts (general/overview articles, specific applications, etc.)

vii) When referring in the text of an article to a term which is the subject of another handbook article, surround the term by +'s; e.g. +Production Systems+.

C. Coding Used In This Outline

This outline contains a list of the major areas of artificial intelligence covered in the handbook. At the lowest level, the outline shows article titles either contained or needed. In the case of an article that is needed, the notation NEED[#] follows the proposed focus of the the article, where # is a number in the interval [0,10]. Low numbers indicate little expected difficulty with the article, whereas high numbers indicate a potentially difficult article. For example, an article on a specific system, where only a minimal amount of reading is required would rate approximately 4, whereas an overview article would likely rate 8 or greater. In the case of articles which already exist in the handbook, the notation done[#] is used, where low numbers indicate that the article needs only minor modifications, and high numbers indicate that major modifications are required. For example, repair of typographical errors and wording could be expected to rate 0-2. Correction of errors in the article might rate 3-6, and major rewrites which require considerable reading would likely rate 7-10.

It should be noted that the real difficulty involved in writing an article is highly dependent on the a priori knowledge of its author.

D. A General View of Artificial Intelligence

Philosophy NEED [9]
This article might address the kinds of questions raised by Turing's article (CAT), Dreyfus's book, the rebuttals, Lighthill's critique, McCarthy's reply, and so on.

Relationship to Society NEED [8]
This might touch on science fiction, popular misconceptions, the Delphi survey, and so on.

History NEED [9]
Perhaps start with Cybernetics, the Dartmouth conference, and so on. See HPS appendix. Also note the major centers, their focus and personalities. Note the role of ARPA funding on the research, the ties to DEC machines and so on.

Conferences and Publications NEED [6]
AI journal, SIGART, SIGCAS, MI books, IJCAI proceedings, CACM, JACM, Cognitive Psychology, some IEEE (Computers, ASSP, SMC), Computational Linguistics, Special interest conferences: robotics, cybernetics, natural language.
Note the tech note unofficial type documents

II. HEURISTIC SEARCH

A. Heuristic Search Overview NEED [9]

Algorithmic presentation of "heuristic search" procedure.

Heuristics for choosing promising nodes to expand next,
heuristics for choosing operators to use to expand a node.

Meta-rules: using heuristics to choose relevant heuristics.

Pervasive character of the combinatorial explosion.

Arguments (both formal and intuitive) supporting the use of heuristic search to muffle this explosion.
Formal: Completeness of A*; Knuth's recent work on alpha-beta search.

Opportunities for future research
Where do heuristics come from?
(see Simon's current work; meta-rules; meta-meta-...?)
Modifying heuristics based on experiences
(see Berliner's current work)
Working with symbolic, rather than numerical, values for nodes
Coding heuristics as production rules
(e.g.: view Mycin as a heuristic search)

Situations NOT suited to attack by heuristic search
Typically: non-exponential growth process; no search anyway
(e.g., finding roots of a quadratic equation)

Identity problems
Disguising Heuristic Search as something else
Disguising something else to appear to be a Heuristic Search

B. Search Spaces

1. Overview — NEED [8]
The concept of a search space; how a search space can be used to solve (some) problems; different representations, different spaces

2. State-space representation — done [6]
[2 articles exist here, which ought to be unified]

3. Problem-reduction representation — done [3]

4. AND-OR trees and graphs — done [4]

C. "Blind" Search Strategies

1. Overview — NEED [5]

2. Breadth-first searching — done [2]

3. Depth-first searching — done [2]

4. Bi-directional searching — NEED [6]

discuss heuristics. MI articles by ira Pohl.

5. Minimaxing — done [3]

6. Alpha-Beta searching — done [3]

D. Using Heuristics to Improve the Search

1. Overview — done [7]
   The idea of a heuristic
   The idea of a heuristic evaluation function
   savings in change of representation.

2. Best-first searching — done [4]
   (Ordered-search) but need to add: Martelli's work (ask Nils for a draft of this)
   speech rec: IJCAI-3 (Paxton), Reddy's book

3. Hill climbing — done [3]

4. Means-ends analysis — done [3]

5. Hierarchical search, planning in abstract spaces — NEED [4]
   Abstrips (Sacerdoti)

6. Branch and bound searching — done [4]

7. Band-width searching — NEED [4]
   Harris - AI journal

E. Programs employing (based on) heuristic search

1. Overview — NEED [7]
   Comparison of systems. Results & limitations.
   (This first article should be written later as an introduction to the following articles.)

2. Historically important problem solvers

   a) GPS — NEED [4]

   b) Strips — NEED [4]

   c) Gelernter's Geom. Program — NEED [3]

III. Natural Language

A. Overview

1. Early machine translation — done [5]

Failures of straight forward approaches

2. History and Development of N.L. NEED [8]
Main ideas (parsing, representation)
comparison of different techniques. mention ELIZA, PARRY.
Include Baseball, Sad Sam, SIR and Student articles here.
see Winograd's Five Lectures, Simmon's CACM articles.

B. Representation of Meaning
(see section VII -- HIP)

C. Syntax and Parsing Techniques

1. overviews
a. formal grammars done [3]
b. parsing techniques NEED [6]

2. augmented transition nets, Woods done [3]

3. Shrdlu's parser (systemic grammars) done [5]

4. Case Grammars Bruce (AI Journal, 1/76) NEED [5]

5. CHARTS - well formed substrings NEED [6]

6. GSP syntax & parser NEED [6]

7. H. Simon - problem understanding NEED [7]

8. transformational grammars done [5]

D. Famous Natural Language systems

1. SHRDLU, Winograd NEED [5]

2. SCHOLAR NEED [5]

3. SOPHIE NEED [5]

E. Current translation techniques NEED [8]
Wilks' work, commercial systems (Vauquois)

F. Text Generating systems NEED [8]
Goldman, Sheldon Klein, Simmons and Sloan (in S&C)

IV. AI Languages

A. Early list-processing languages done [3]

overview article

languages like COMIT, IPL, SLIP, SNOBOL, FLPL

Ideas: recursion, list structure,
associative retrieval

B. Language/system features

0. Overview of current list-processing languages — NEED [7]

1. Control structures, what languages they are in and examples of their use. — NEED [6]

   Backtracking (parallel processing)
   Demons (pseudo-interrupts)
   Pattern directed computation

2. Data Structures (lists, associations, bags, tuples, property lists,...) — NEED [5]

   Once again, examples of their use is important here.

3. Pattern Matching in AI languages — NEED [6]
   see Bobrow & Raphael

4. Deductive mechanisms — NEED [5]
   see Bobrow & Raphael

C. Current languages/systems

1. LISP, the basic idea — done [2]

2. INTERLISP — NEED [5]

3. QLISP (mention QA4) — done [3]

4. SAIL/LEAP — done [2]

5. PLANNER — done [2]

6. CONNIVER — done [2]

7. SLIP — NEED [4]

8. POP-2 — NEED [4]

9. SNOBOL — NEED [4]

10. QA3/PROLOGUE — (see thm. prov.)

V. AUTOMATIC PROGRAMMING

A. Overview done [7]
B. Program Specification Techniques
i.e. how does the user describe the program to be synthesized?

--an overview article including various methods NEED[9]
see SAFE system (ISI), Green's tech. report, DSSL, Smith's graphic specification, and include general remarks on the high-level language methods

C. Program Synthesis techniques NEED[9]
- given a description of the program in some form, generate the actual program

1. Traces done[3]

2. Examples done[3]
(include Biggerstaff at U. of Washington)

3. Problem solving applications to AP NEED[9]
--including classical problem-solving techniques, plan modification, "pushing assertions across goals," and theorem proving techniques. (debugging (Sussmans's Hacker), Simon's Heuristic Compiler, and Prow (Waldinger) & QA3)
(Should Theorem-Proving-Techniques remain a separate article?)

4. Codification of Programming Knowledge NEED[?]
see C.Green's work, Darlington, Rich & Shrobe

5. Integrated AP Systems NEED[?]
see Lenat's original work, Heidorn, Martin's OWL, PSI at SAIL

D. Program optimization techniques NEED [7]
How to turn a rough draft into an efficient program. See Darlington & Burstall, Low, Wegbreit, Kant.

E. Programmer's aids NEED [7]
(Interlisp's DWIM, etc)

F. Program verification NEED [7]
(IJCAI 3)

VI. THEOREM PROVING

A. Overview NEED [9]

B. Resolution Theorem Proving

1. Basic resolution method done [4]

2. Syntactic ordering strategies — done [2]

3. Semantic & syntactic refinement — done [2]

[4. other strategies?]

C. Non-resolution theorem proving

1. Natural deduction — done [3]

2. Boyer-Moore

3. LCF

D. Uses of theorem proving

1. Use in question answering — done [3]

2. Use in problem solving — done [6]

3. Theorem Proving languages (QA3, Prologue) — NEED [5]

4. Man-machine theorem proving (Bledsoe) — NEED [6]

E. Predicate Calculus — done [5]

F. Proof checkers

VII. Human Information Processing -- Psychology

(see Perry's outline for details and references)

A. Perception — NEED [9]

An overview of relevant work in psychology on attention, visual and auditory perception, pattern recognition. Applied perception (PERCEIVER). Difficulties resulting from inability to introspect.

B. Memory and Learning

1. Basic structures and processes in IPP — NEED [9]

Short- and Long-term memory, Rehearsal, Chunking, Recognition, Retrieval, recall, Inference and question-answering, Semantic vs. episodic memory, Interference and forgetting, Type vs. token nodes
Simon - Sciences of the Artificial

2. Overview of memory models, Representation — NEED [10]

How to get to the airport: A comparison of the various models.

a. Associative memory models

1. semantic nets — NEED [9]
   Quillian (TLC), Nash-Weber (BBN) Shapiro, Hendricks (SRI), Wood's article in Bobrow & Collins, Simmons (S&C)

2. HAM (Anderson & Bower) — NEED [7]

3. LNR: Active Semantic Networks — NEED [6]

4. Componential analysis — NEED [9]
   Jakendoff, Schank (conceptual dependency), (MARGIE), G. Miller

5. EPAM — NEED [5]

6. Query languages — NEED [7]
   Wood's (1968), Ted Codd (IBM SJ)

b. Other representations

1. Production systems — done [1]

2. Frame systems (Minsky, Winograd) — done [7]

3. Augmented Transition Networks — done [3]

4. Scripts (Schank, Abelson) — NEED [7]

C. Psycholinguistics — NEED [9]

A prose glossary including:
Competence vs. performance models, Phonology vs. syntax vs. semantics vs. pragmatics, Surface vs. deep structure, Taxonomic grammars, generative grammars, transformational grammars, Phrase-structure rules, transformation rules, Constituents, lexical entries Parsing vs. generation, Context-free vs. Context-sensitive grammars, Case systems (e.g., Bruce AI article)

D. Human Problem Solving -- Overview — NEED [8]

1. PBG's — done [1]

2. Concept formation (Winston) — done [2]

3. Human chess problem solving — NEED [6]

E. Behavioral Modeling

1. Belief Systems — NEED [8]
   Abelson, McDermott

2. Conversational Postulates (Grice, TW) — NEED [5]

3. Parry — NEED [5]

VIII. VISION

A. Overview — NEED [9]

This article should discuss the early work in vision; its roots in pattern recognition, character recognition, Pandemonium, Perceptrons and so on. (i.e.. the pre-Roberts work). It should discuss the main ideas of modern vision work as a leadin to the more specific articles, for example the use of hypothesis, model, or expectation driven strategies. It should also discuss the way in which the focus of the field flip-flops from front end considerations to higher level considerations with time.

B. Polyhedral or Blocks World Vision

An overview article should include the major ideas in this work together with brief summaries of the work of the major investigators. In addition, separate articles should be written on the work of those listed below.

Overview — NEED [7]
(Roberts, Huffman and Clowes, Kelley, Shirai and others listed below)

Guzman — done [2]

Falk — NEED [5]

Waltz — NEED [7]
This article should contain more general material on constraint satisfaction, drawn possibly from Montenari and Fikes

This exhausts my list. Please add others or delete some of mine if appropriate.
It has been suggested [Bolles] that the most instructive method of writing these articles would be to provide simple examples of the problems attacked by the various programs.

C. Scene Analysis

Overview NEED [9]
This article should describe or point to detailed strategies used, and the present state of the art.

The following articles should be written or modified to describe the specialized tools of scene analysis. See Duda and Hart.

Template Matching NEED [5]
(a non-mathematical description)

Edge Detection done [4]

Homogeneous Coordinates done [7]
This article should be modified to include the general questions of the perspective transformation, camera calibration, and so on.

Line Description done [4]

Noise Removal done [4]

Shape Description done [4]

Region Growing (Yakamovsky, Olander) done [3]

Contour Following NEED [4]

Spatial Filtering NEED [4]

Front End Particulars NEED [6]
This article should contain some description of the methods and effects of compression and quantization for example.

Syntactic Methods NEED [5]

Descriptive Methods NEED[6]
See Duda and Hart, and Winston

D. Robot and Industrial Vision Systems

Overview and State of the Art NEED [9]

Hardware NEED [8]

E. Pattern Recognition

It's not clear just where this discussion should go, or what level of detail is required.

Overview done [8]

This article needs to be refocussed and cleaned up

Statistical Methods and Applications NEED [9]

Descriptive Methods and Applications NEED [8]

F. Miscellaneous

Multisensory Images NEED [7]

Perceptrons NEED [6]

IX. SPEECH UNDERSTANDING SYSTEMS

Overview (include a mention of ac. proc.) done [3]

Integration of Multiple Sources
of Knowledge NEED [9]
For example the blackboard of the HEARSAY II system

HEARSAY I done [4]

HEARSAY II done [5]

SPEECHLIS done [2]

SDC-SRI System (VDMS) NEED [7]

DRAGON done [6]
Jim Baker's original system plus Speedy-Dragon by Bruce Lowerre. This article is a little harder than the other system articles because the methods used may be unfamiliar to some.

X. ROBOTICS

Overview NEED [9]
This article should discuss the central issues and difficulties of the field, its history, and the present state of the art.

Robot Planning and Problem Solving NEED [8]
For example, STRIPS and ABSTRIPS. This article could be quite general depending on the point of view taken.

Arms NEED [8]
Explain the difficulties of control at the bottom level, system integration, obstacle avoidance and so on. Also note the problems with integration of multi-sensory data, for example vision and

touch feedback.

Present Day Industrial Robots NEED [7]

Robotics Programming Languages NEED [6]
For example WAVE, and AL
(a short article)

XI. Applications of AI

An overview article. What are the attributes of a suitable domain? Custom crafting - theory vs. actual use. (See EAF: 225 notes, 1972) NEED [8]

A. Chemistry

1. Mass spectrometry (DENDRAL, CONGEN, meta-dendral) done [6]

2. Organic Synthesis
Overview NEED [8]
Summarize work of Wipke, Corey, Gelernter, and Sridharan

B. Medicine

1. MYCIN done[1]

2. Summarize DIALOG(Pople), CASNET(Kulikowski), Pauker's MIT work, and the Genetics counselling programs NEED[7]

C. Psychology and Psychiatry

Protocol Analysis (Waterman and Newell) NEED [6]

D. Math systems

1. REDUCE NEED [4]

2. MACSYMA (mention SAINT) NEED [6]

E. Business and Management Science Applications

1. Assembly line balancing (Tonge) NEED [5]

2. Electric power distribution systems (MI) NEED [5]

F. Miscellaneous

1. LUNAR NEED [5]

2. Education NEED [7]
Papert, or more ?

3. SRI computer-based consultation	NEED [6]
4. RAND--RITA production rule system for intelligent interface software	NEED [5]

I. Miscellaneous

Overview of music composition and aesthetics	done [7]

XII. Where do these go?

Reasoning by analogy	done [4]
Intelligence augmentation	done [5]
Chess	done [5]

XIII. Learning and Inductive Inference

Overview	NEED [9]
Samuel Checker program	NEED [5]
Winston	done [2]
Pattern extrapolation problems--Simon, Overview of Induction	NEED [5]

APPENDIX C

HEURISTIC PROGRAMMING PROJECT WORKSHOP

In the first week of January 1976, about fifty representatives of local SUMEX-AIM projects convened at Stanford for four days to explore common interests. Six projects at various degrees of development were discussed during the conference. They included the DENDRAL and META-DENDRAL projects, the MYCIN project, the Automated-Mathematician project, the Xray-Crystallography project, and the MOLGEN project. Because of the interdisciplinary nature of each of these projects, the first day of the conference was reserved for tutorials and broad overviews. The domain-specific background information for each of the projects was presented and discussed so that more technical discussions could be given on the following days. In addition the scope and organization of each of the projects was presented focusing on the tasks that were being automated, how people perform these tasks, and why the automation was useful or interesting.

In the following days of the workshop, common themes in the management and design of large systems were explored. These included the modular representations of knowledge, gathering of large quantities of expert knowledge, and program interaction with experts in dealing with the knowledge base. Several of the projects were faced with the difficulties of representing diverse kinds of information and with utilizing information from diverse sources in proceeding towards a computational goal. Parallel developments within several of the projects were explored, for example, in the representation of molecular structures and in the development of experimental plans in the MOLGEN and DENDRAL projects. The use of heuristic search in large, complex spaces was a basic theme to most of the projects. The use of modularized knowledge typically in the form of rules was explored for several of the projects with a view towards automatic acquisition, theory formation, and program explanation systems.

For each of the projects, one session was devoted to plans for future development. One of the interesting questions for these sessions was the effect of emerging technology on feasibility of new aspects of the projects. The potential uses of distributed computing and parallel processing in the various projects were explored, particularly in the context of the DENDRAL project.

Most of the participants felt that the conference gave them a better understanding of related projects. And because many members of the SUMEX-AIM staff actively participated, the workshop also provided all projects with information about system developments and plans. The discussions and sharing of ideas encouraged by this conference has continued through a series of weekly lunches open to this whole community.

APPENDIX D

TYMNET RESPONSE TIME DATA

Following are statistics on one-way character transit time delays over the TYMNET derived from the collected TYMSTAT data between June 1975 and April 1976. The first line in each section contains the node ID. Then for each month when data were available for that node, the succeeding tables in the section give the number of data points collected and delay statistics in milliseconds for various parts of the day (Pacific Time). These data have been the basis of numerous conversations with TYMSHARE over the past year attempting to correct intolerable delay times. That fight goes on!

An index to particular nodes follows:

1010 OAKLAND CALIFORNIA OAK1 E ** 415/465-7000

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	1			
Average Delay	282.0			
Std Deviation	.0			
Minimum Delay	282			
Maximum Delay	282			

August 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00

Number	1
Average Delay	365.0
Std Deviation	.0
Minimum Delay	365
Maximum Delay	365

1011 WASHINGTON D.C. WASSR1 C ** 703/841-9560

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		1		
Average Delay		204.0		
Std Deviation		.0		
Minimum Delay		204		
Maximum Delay		204		

September 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		5		
Average Delay		177.6		
Std Deviation		38.9		
Minimum Delay		123		
Maximum Delay		227		

October 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	1			
Average Delay	153.0			
Std Deviation	.0			
Minimum Delay	153			
Maximum Delay	153			

November 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	2			
Average Delay	144.5			
Std Deviation	13.5			
Minimum Delay	131			
Maximum Delay	158			

1012 CHICAGO ILLINOIS CHI2 E ** 312/346-4961

December 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	1	3		
Average Delay	346.0	393.0		
Std Deviation	.0	160.8		
Minimum Delay	346	214		
Maximum Delay	346	604		

March 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00

Number	2
Average Delay	251.5
Std Deviation	66.5
Minimum Delay	185
Maximum Delay	318

1014 MIDLAND TEXAS MDL1 C 915/683-5645

June 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	2			1
Average Delay	525.0			310.0
Std Deviation	158.0			.0
Minimum Delay	367			310
Maximum Delay	683			310

1017 PALO ALTO CALIFORNIA PA1 C ** 415/494-3900

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number			1	
Average Delay			414.0	
Std Deviation			.0	
Minimum Delay			414	
Maximum Delay			414	

1022 WASHINGTON D.C. WAS2 E ** 703/521-6520

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		3		
Average Delay		188.0		
Std Deviation		10.6		
Minimum Delay		173		
Maximum Delay		196		

September 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		3		
Average Delay		197.3		
Std Deviation		23.5		
Minimum Delay		165		
Maximum Delay		220		

October 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		2		
Average Delay		261.0		
Std Deviation		3.0		
Minimum Delay		258		
Maximum Delay		264		

November 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	3	12		
Average Delay	242.7	300.4		
Std Deviation	64.5	161.8		
Minimum Delay	153	129		
Maximum Delay	302	774		

December 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		2		
Average Delay		208.0		
Std Deviation		49.0		
Minimum Delay		159		
Maximum Delay		257		

1023 SEATTLE WASHINGTON SEA1 C 206/622-7930

September 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	1	1		
Average Delay	385.0	391.0		
Std Deviation	.0	.0		
Minimum Delay	385	391		
Maximum Delay	385	391		

March 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		1		
Average Delay		805.0		
Std Deviation		.0		
Minimum Delay		805		
Maximum Delay		805		

1027 LOS ANGELES CALIFORNIA LA2 E ** 213/683-0451

December 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		2		
Average Delay		162.0		
Std Deviation		6.0		
Minimum Delay		156		
Maximum Delay		168		

January 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	3			
Average Delay	172.3			
Std Deviation	9.4			
Minimum Delay	161			
Maximum Delay	184			

1034 NEW YORK NEW YORK NYCSR1 E ** 212/532-7615
NEW YORK NEW YORK NYCSR1 E ** 212/551-9322

June 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		8		
Average Delay		561.9		
Std Deviation		98.9		
Minimum Delay		407		
Maximum Delay		709		

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	3	7		
Average Delay	511.3	518.3		
Std Deviation	53.8	105.3		
Minimum Delay	458	407		
Maximum Delay	585	732		

September 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	2	15		
Average Delay	418.0	365.9		
Std Deviation	95.0	187.7		
Minimum Delay	323	187		
Maximum Delay	513	828		

October 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	13	15		
Average Delay	712.2	394.3		
Std Deviation	523.5	147.2		
Minimum Delay	335	182		
Maximum Delay	1783	768		

November 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	19	21		
Average Delay	635.4	380.6		
Std Deviation	511.0	55.4		
Minimum Delay	224	264		
Maximum Delay	2183	510		

December 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	13	33		
Average Delay	855.2	931.2		
Std Deviation	996.8	908.4		
Minimum Delay	190	223		
Maximum Delay	2763	3035		

January 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	4	11		
Average Delay	466.0	591.4		

Std Deviation	152.7	180.0
Minimum Delay	226	233
Maximum Delay	621	901

February 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	2	11		
Average Delay	508.5	709.7		
Std Deviation	53.5	160.3		
Minimum Delay	455	466		
Maximum Delay	562	1028		

March 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	8	5		
Average Delay	849.8	581.8		
Std Deviation	315.1	230.1		
Minimum Delay	487	331		
Maximum Delay	1351	953		

April 1976

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	13	6		
Average Delay	1180.4	794.3		
Std Deviation	511.8	304.0		
Minimum Delay	529	471		
Maximum Delay	2108	1346		

1036 NEW YORK NEW YORK NY1 E ** 212/344-7445

June 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	4	3		
Average Delay	687.8	495.3		
Std Deviation	66.9	134.8		
Minimum Delay	609	339		
Maximum Delay	756	668		

July 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number	6	1		
Average Delay	426.5	847.0		
Std Deviation	77.2	.0		
Minimum Delay	338	847		
Maximum Delay	562	847		

September 1975

	05:00-09:00	09:00-17:00	17:00-22:00	22:00-05:00
Number		4		
Average Delay		380.8		
Std Deviation		34.0		
Minimum Delay		346		
Maximum Delay		428		